AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (Original) In a computational system wherein a value of a lock is encoded to identify (i) a set of one or more transactions that own the lock and (ii) respective one or more modes in which such transactions own the lock, and wherein at least some locks of equal value are represented using a same shared lock state, a method of performing bulk delegation of locks from one or more delegator transactions to one or more delegatee transactions, the method comprising:

traversing a data structure encoding the shared lock states; and

for each shared lock state of the data structure encoding having an owner set that includes at least one of the delegator transactions, removing from the owner set each of the delegator transactions, and adding thereto each of the delegatee transactions.

2. (Original) A method, as recited in claim 1,

wherein the shared lock states include a respective owner set per lock mode such that each owner set encodes a set of the transactions that owns the lock in the respective lock mode.

3. (Previously Presented) A method, as recited in claim 2,

wherein the removing and the adding are performed for each of the owner sets that includes at least one of the delegator transactions.

4. (Original) A method, as recited in claim 1,

wherein the set of the shared lock states is at least partially encoded in an associative table of shared lock states (TSLS).

5. (Original) A method, as recited in claim 4,

wherein at least a subset of the shared lock states that represent locks owned by a single owner is encoded separately from the table of shared lock states (TSLS).

6. (Original) A method, as recited in claim 4,
wherein at least a frequently used subset of the shared lock states is encoded separately from the table of shared lock states (TSLS).

7. (Original) A method, as recited in claim 1, wherein the bulk delegation is one of:
a one-to-many bulk delegation;
a many-to-one bulk delegation; and
a many-to-many bulk delegation.

8. (Original) A method, as recited in claim 1, further comprising:
selecting a subset of the locks for the bulk delegation based at least in part on a mode or modes in which the subset of the locks is held.

9. (Original) In a computational system wherein at least some lock states are shared amongst transactions, a method of implementing a bulk delegation of locks from one or more delegator transactions to one or more delegatee transactions, the method comprising:
scanning an encoding of shared lock states; and
for each shared lock state encoding having at least one associated owner set that includes one of the delegator transactions, removing from the associated owner set each of the delegator transactions and adding thereto each of the delegatee transactions.

10. (Original) A method, as recited in claim 9,
wherein at least some of the shared lock state encodings have plural associated owner sets, each corresponding to a respective mode in which a corresponding lock can be owned.

11. (Previously Presented) A method, as recited in claim 9,
wherein the removing and the adding are performed for each of the plural owner sets that includes one of the delegator transactions.

12. (Original) The method of claim 9, further comprising:
prior to the removing and adding, validating the bulk delegation based at least in part on ignore conflicts relationships amongst the delegatee transactions.

13. (Previously Presented) The method of claim 9, further comprising:
prior to the removing and adding, validating the bulk delegation based, at least in part, on ignore conflicts relationships amongst the delegatee transactions and between the delegatee transactions and otherwise incompatible-mode owners of locks, which would remain after completion of the bulk delegation.

14. (Original) The method of claim 9,
wherein the locks are held or owned in modes that at least include read and write modes.

15. (Original) The method of claim 9,
wherein the locks are held or owned in modes selected from a set of lock modes; and
wherein at least some individual ones of the lock modes encompass other ones of the lock modes based on a precedence relation.

16. (Original) The method of claim 9,
wherein the encoding of shared lock states includes an associative search data structure.

17. (Original) The method of claim 16,
wherein at least some frequently used lock states are encoded apart from the associative search data structure.

18. (Original) The method of claim 16,
wherein at least some single owner lock states are encoded apart from the associative search data structure.

19. (Original) The method of claim 9,
wherein the locks are delegated from a single delegator transaction to plural delegatee transactions.

20. (Original) The method of claim 9,
wherein the locks are delegated from plural delegator transactions to a single delegatee transaction.

21. (Previously Presented) The method of claim 9,
wherein the locks are delegated from plural delegator transactions to plural delegatee transactions.

22. (Original) A transaction processing system that supports bulk delegation of locks, the transaction processing system comprising:
a lock manager that associates locking capabilities with transactions and that allows specification of certain conflicts between locking capabilities to be ignored; and
an encoding of shared lock states, the lock manager implementing a bulk delegation operation by scanning the shared lock states and, for each shared lock state having an associated owner set that includes one of the delegator transactions, removing from the associated owner set each of the delegator transactions and adding thereto each of the delegatee transactions.

23. (Previously Presented) The transaction processing system of claim 22,
wherein at least some of the shared lock states have plural associated owner sets corresponding to respective lock modes; and
wherein the removing and adding are performed on each of the owner sets that include at least one delegator transaction.

24. (Original) The transaction processing system of claim 22,
wherein the transaction processing system implements a nested transaction model; and
wherein the bulk delegation operation is employed to support creation, commitment or abortion of a sub-database.

25. (Original) The transaction processing system of claim 22,
wherein the lock manager validates a bulk delegation request based, at least in part, on ignore conflicts relationships amongst the delegatee transactions.

26. (Original) The transaction processing system of claim 22, wherein the lock manager:

verifies that, for each type of conflict between lock modes, a set of transactions with which a particular delegatee transaction can ignore conflicts is a super-set of the set of transactions with which the delegator transactions can ignore conflicts; and

if at least one to-be-delegated lock is held in a write mode by at least one of the delegator transactions, further verifies that each of the delegatee transactions can ignore conflicts with each other.

27. (Currently Amended) A computer implemented lock manager that implements a bulk delegation operation by validating a bulk delegation request based, at least in part, on ignore conflicts relationships amongst delegatee transactions, and for a validated bulk delegation request, by (i) scanning an encoding of shared lock states and (ii) for each shared lock state having an associated owner set that includes a delegator transaction, removing therefrom each of the delegator transactions and adding thereto each delegatee transactions~~, wherein the lock manager is embodied as software executable in an transaction processing environment~~.

28. (Currently Amended) The lock manager of claim 27, wherein the lock manager is executable in a transaction processing environment that associates locking capabilities with transactions and that allows specification of certain conflicts between locking capabilities to be ignored.

29. (Original) A computer program product encoded in one or more computer readable media and comprising:

definition of a data structure instantiable in memory to represent plural locks having identical lock values using a single shared lock state encoding;

lock manager instructions executable by a processor to associate locking capabilities with transactions, to specify certain conflicts between locking capabilities to be ignored, to manage the shared lock state encoding, and to implement a bulk delegation of locks from one or more delegator transactions to one or more delegatee transactions, the bulk delegation scanning the shared lock state encoding and, for each shared lock state having an associated owner set that

includes one of the delegator transactions, removing therefrom each of the delegator transactions and adding thereto each of the delegatee transactions.

30. (Original) The computer program product of claim 29,
wherein the lock manager instructions validate a bulk delegation request based, at least in part, on ignore conflicts relationships amongst the delegatee transactions.

31. (Original) The computer program product of claim 29,
wherein the one or more computer readable media are selected from the set of a disk, tape or other magnetic, optical or electronic storage medium and a network, wireline, wireless or other communications medium.

32. (Original) An apparatus comprising:
means for representing plural locks having identical lock values using a single shared lock state encoding; and
means for delegating in bulk locks from one or more delegator transactions to one or more delegatee transactions based, at least in part, on ignore conflicts relationships amongst the delegatee transactions.

33. (Previously Presented) The lock manager of claim 27, wherein the lock manager is encoded in one or more computer readable media.

34. (Previously Presented) The lock manager of claim 33, wherein the computer readable media includes one or more of magnetic storage medium, optical storage medium, magneto-optical storage medium, read only memory, random access memory, erasable programmable memory, flash memory, and propagated signal.